Singularity

Galen Hunt
James Larus
David Tarditi

Microsoft Research

MSR TAB
June 21, 2005

General-purpose Computing Platform (PC)

- Software is difficult to install, maintain, and administer
- Applications interact in complex, unpredictable ways
- Almost no users understand computers or software and so react naively to unexpected behavior
- System administration is costly and unavailable to most

Talk Summary

- Advances in languages, compilers, and tools open the possibility of improving software
- Singularity uses these advances as a basis to build more reliable systems and applications
- Systems built on Singularity expand software delivery opportunities

Advances in Language & Compiler Technology

- Software advances enable better system architectures
  - more robust and verifiable than existing (40 year old) model

	Advance	
	Language safety	**Program and system verification**
Technology	expressive safe languages and type systems	explicit specifications and system descriptions
	optimizing compilers	defect detection/testing tools
	end-to-end safety checking	static program analysis
[illegible]	increasingly precise analysis	improved static and dynamic error detection
	systems with finer-grain isolation and fault-containment boundaries	strong system guarantees

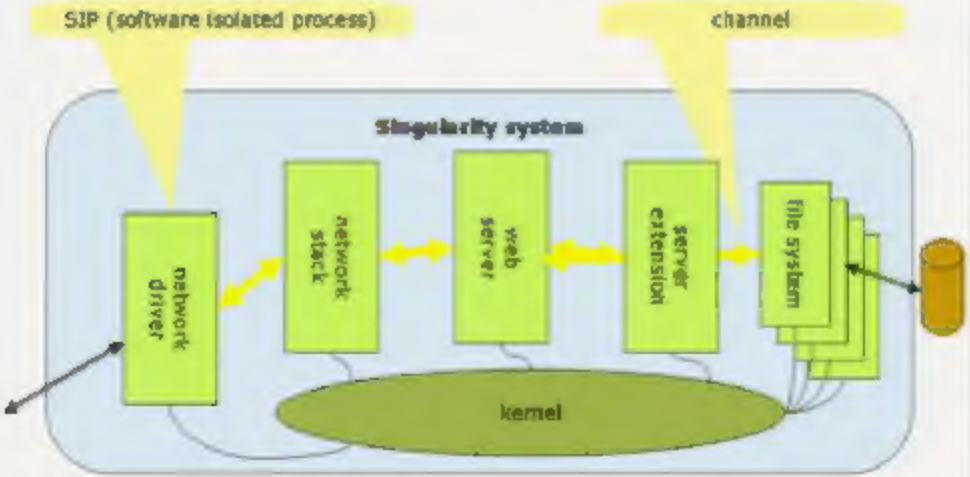
Singularity Project

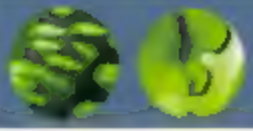
- Develop technology and infrastructure to build more dependable software
  - language support to increase software quality
  - tools to ensure correct software behavior
  - OS architecture to enhance security, reliability, and effectiveness of tools
  - superior reliability, sufficient performance
- Large research project
  - current: ~dozen researchers & RSDEs
  - Redmond, SVC, Cambridge
- System running on hardware and Virtual PC
- Not Windows successor!

Key Aspects of Singularity

- Software-isolated processes (SIPs)
  - Inexpensive isolation — memory, communications, failure — boundaries
  - OS manages and reclaims resources
  - "closed world" for program analysis
  - single isolation and extension model for all parts of application and system
- Merge OS and language runtime (VM/CLR)
  - prohibit unsafe code
  - remove duplicative APIs and security abstractions
  - fast, lightweight managed code run-time system
  - typed assembly language (TAL) reduces trusted computing base
- Language extensions to improve reliability
  - Spec#/Sing# specifications and verification
  - channel contracts
  - explicit and verified resource usage and reclamation

Singularity Architecture



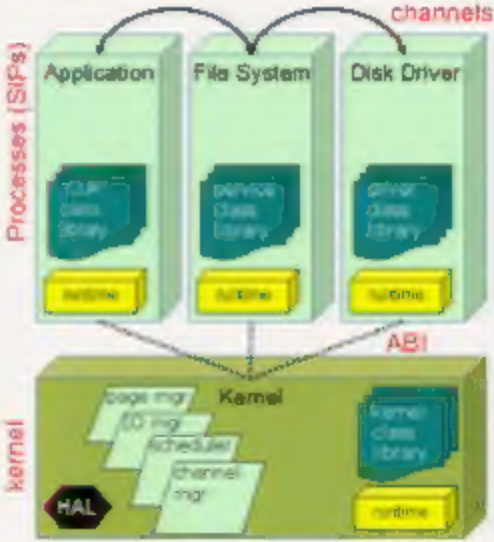


Presentation

- Introduction (3m)
- Singularity OS (Galen)
- Language, compiler & runtime (David)
- Opportunities (3m)

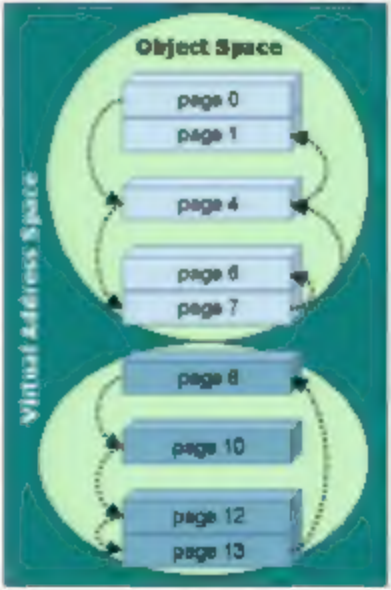
Detailed Architecture

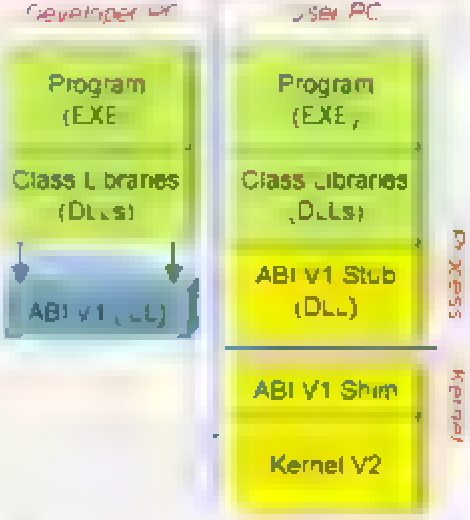
- Microkernel
  - apps, extensions, services, and drivers are all processes
  - HAL w/ PIC, RTC, timer, and console output
- Closed processes (SIPs)
  - no shared memory
  - no dynamic code loading
  - no dynamic code generation
- IPC via channels w/ contracts
- Abstract instruction set
  - type safe, memory safe MSIL
  - all third-party code is safe
  - layer of indirection
- Well-defined, strongly-versioned application binary interface (ABI)



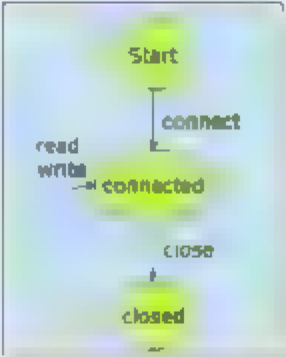
Software Isolated Processes (SIPs)

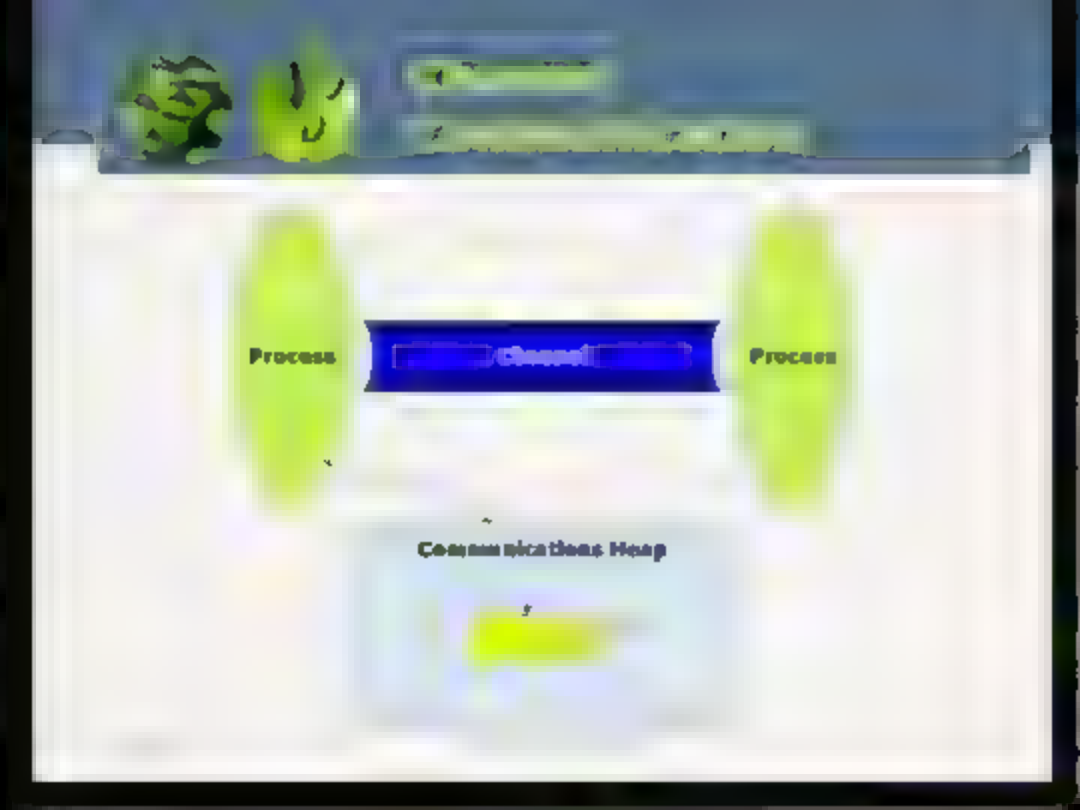
- Each SIP
  - contains verified user code
    - type safe
    - no privileged instructions
  - has exclusive ownership of pages
  - contains pointers to its pages only
    - self-contained for GC
  - has own runtime
- SIPs can
  - run in kernel address space
  - run in ring0
    - kernel hw protection domain



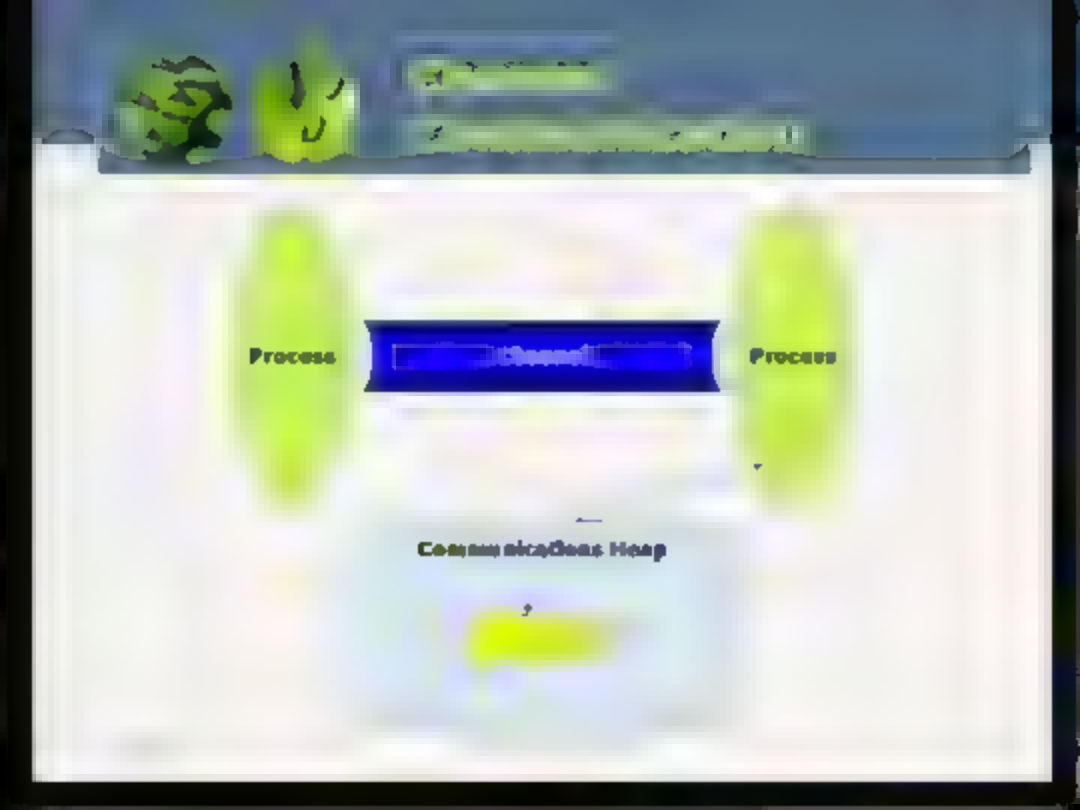

Program
Class Libraries
ABI V1 Stub
(DLL)
ABI V1 Shim
Kernel V2

- Channels
  - asynchronous & [illegible]
  - exactly two endpoints
  - [illegible]
    - [illegible]
    - [illegible]
- Messages may contain values and endpoints
- Messages and endpoints [illegible]
  - send removes message from sender
  - receive transfers message to receiver
  - at most one owning [illegible] at any time




Process
Channel
Process
Communications Heap

Communications Heap

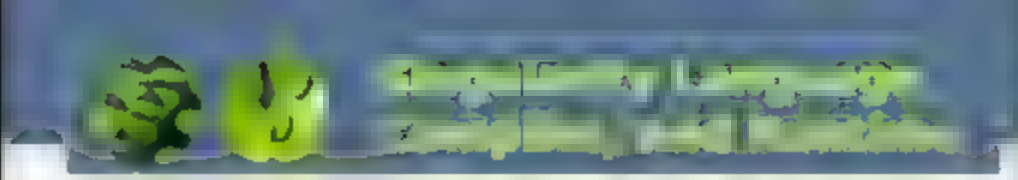

Process
Channel
Process
Communications Heap

Cost (CPU Cycles)

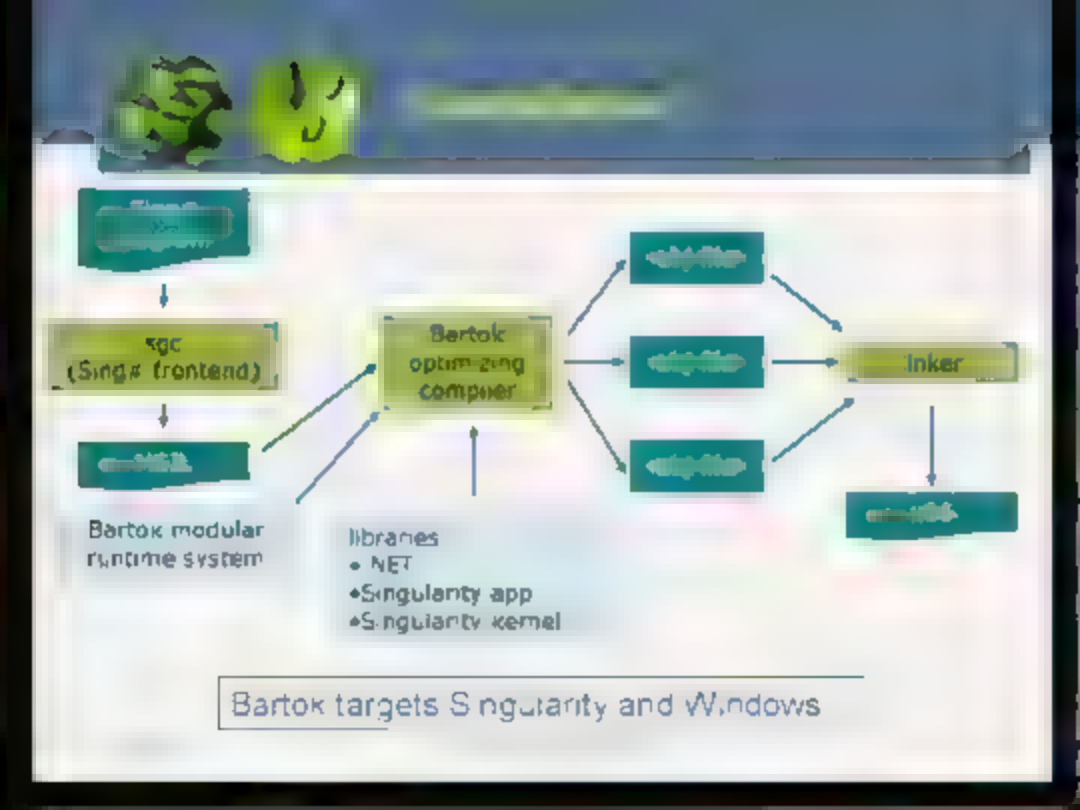
	Singularity	FreeBSD	Linux	Windows
A [illegible]	[illegible]	[illegible]	[illegible]	4 [illegible]
M [illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	1,[illegible]	3,41[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

- Bartok compiler and runtime
  - [illegible]
  - lightweight [illegible]
- Singularity kernel
  - processes and channels supported
  - drivers and services [illegible]
- [illegible]
  - [illegible]
  - additional [illegible] development
- [illegible] functionality
  - [illegible]
  - nameserver and [illegible] manager
  - Bo[illegible] B-tree [illegible] system

- Introduction
- Singularity OS
- Language, compiler & runtime
- Opportunities



- Systems programming in safe, garbage collected language
- Built on Bartok compiler & run time system and Spec# language
- Language extensions
  - [illegible]
  - [illegible]
- Highly optimizing, ahead of time compiler
- Small, customizable run time system
  - [illegible]



Bartok targets Singularity and Windows

- System integrity depends on code safety
  - Type and memory safety [illegible]
- Deter[illegible] [illegible]

- Current[illegible] [illegible] compiler ensures [illegible]
- Working on [illegible] assembly language [illegible] TCB

Supported

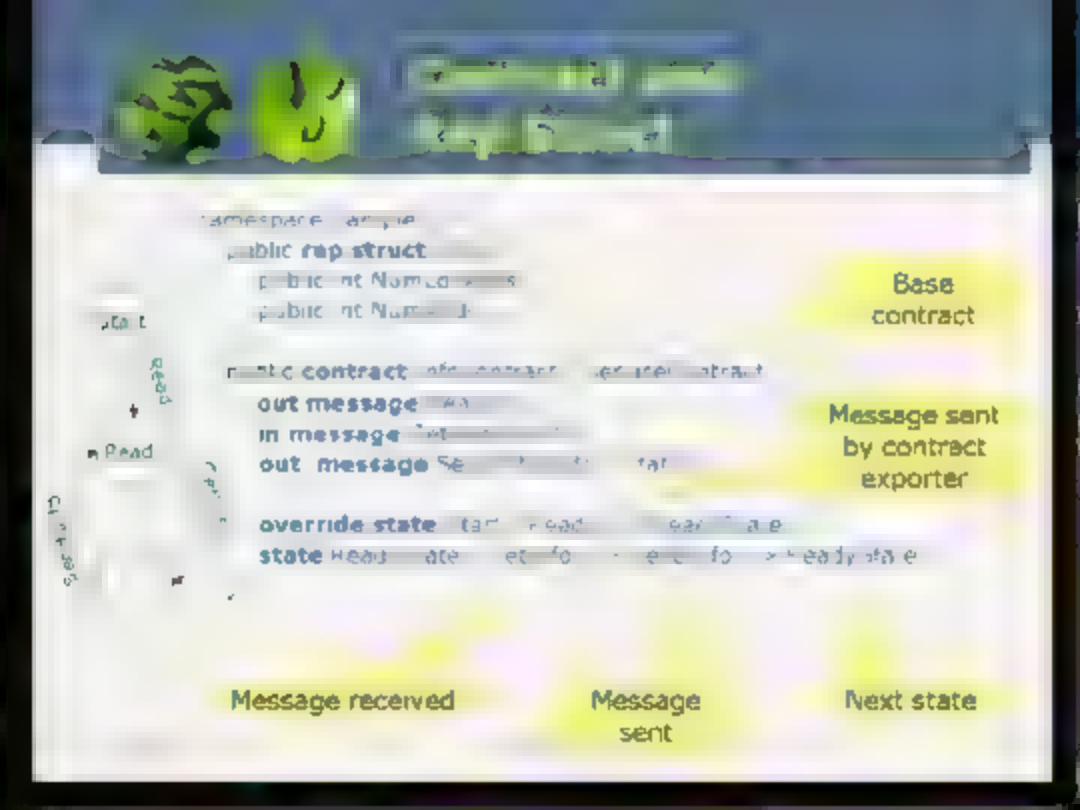
- Value types + [illegible]
    - [illegible] managed [illegible]
- Garbage collection [illegible] managers [illegible]
- Exception handling
- Lazy type initialization
- Delegates
- Unsafe code [illegible]
- Data about attributes
- Generics [illegible]

Not supported

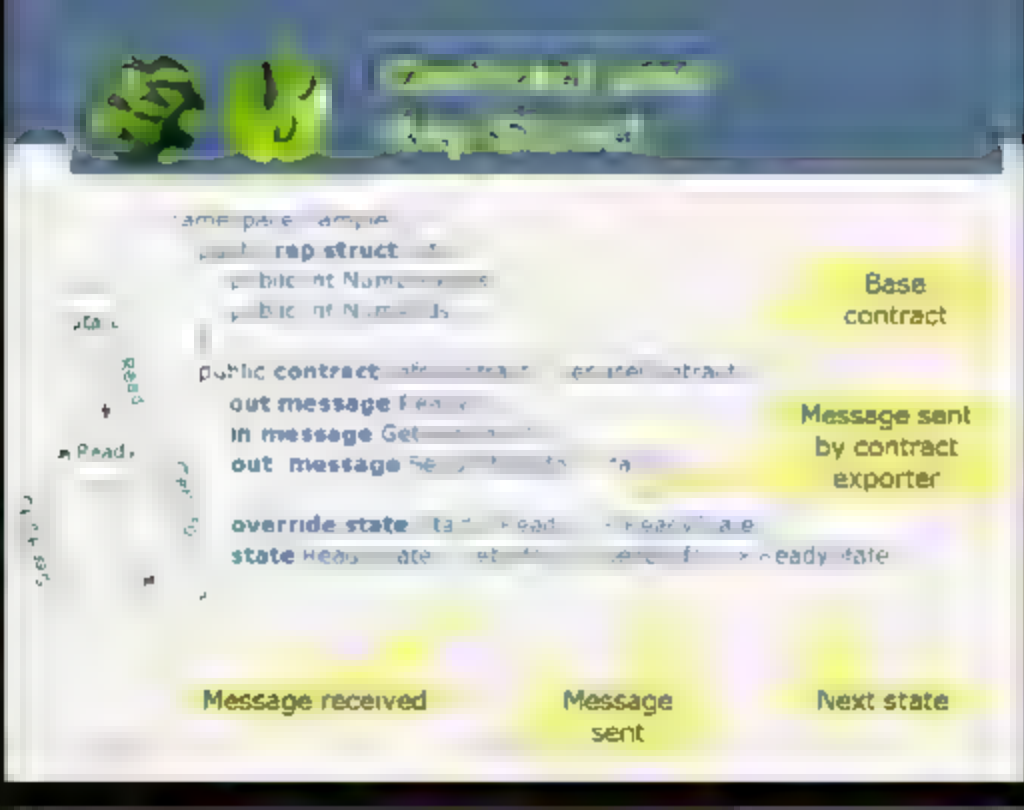
- Dynamic class loading
- Reflection: very limited support
- Code access security
- Platform invoke

- [illegible]
- [illegible]
  - [illegible]
  - [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible]

- Problem: efficient communications without data sharing
- Answer: [illegible]
  - data transferred from sender to receiver
  - sender [illegible] message [illegible]
  - permits [illegible]
- Enforced by type system
- [illegible]
  - [illegible]
  - pointer [illegible]
  - pointers [illegible]
- [illegible]
  - allocated outside of heap
  - [illegible]
  - cannot be used after [illegible] message
- Rep structs and primitive types are only types passed between processes


Base contract
Message sent by contract exporter
Message received
Message sent
Next state

- Contract declares
  - message types
  - state machine
- Provides
  - typed
  - handler
  - typed
- Efficient
  - pre-
  - state


rep struct
Base contract
Message sent by contract exporter
Message received
Message sent
Next state

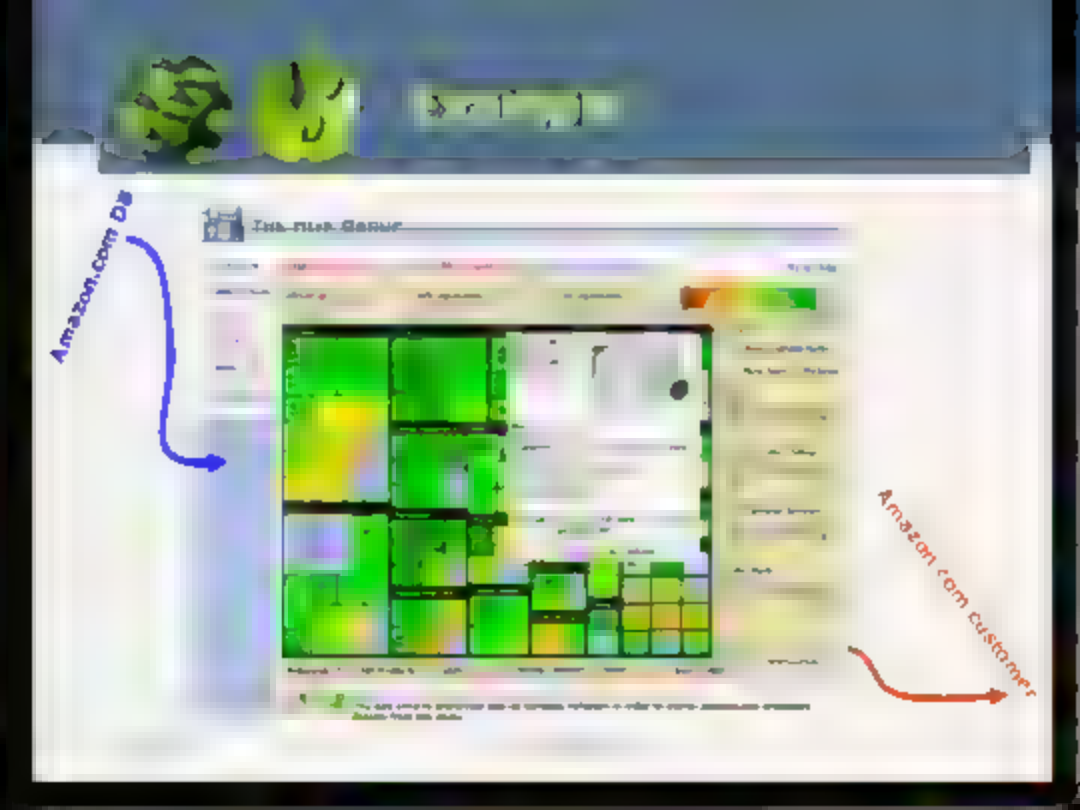
- Bartok supports whole program and separate compilation
- Assists specialization of runtime systems
- Extensive optimization

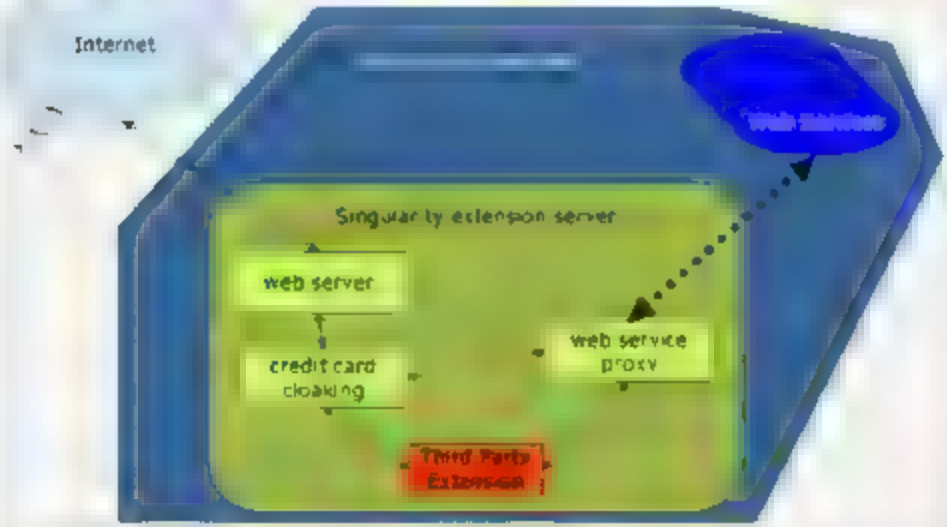
- Each process run-time environment is autonomous
  - [illegible]
- Customize environment for its code
  - [illegible]
    - [illegible]
    - [illegible]
  - [illegible]
    - [illegible]
  - [illegible]
    - [illegible]

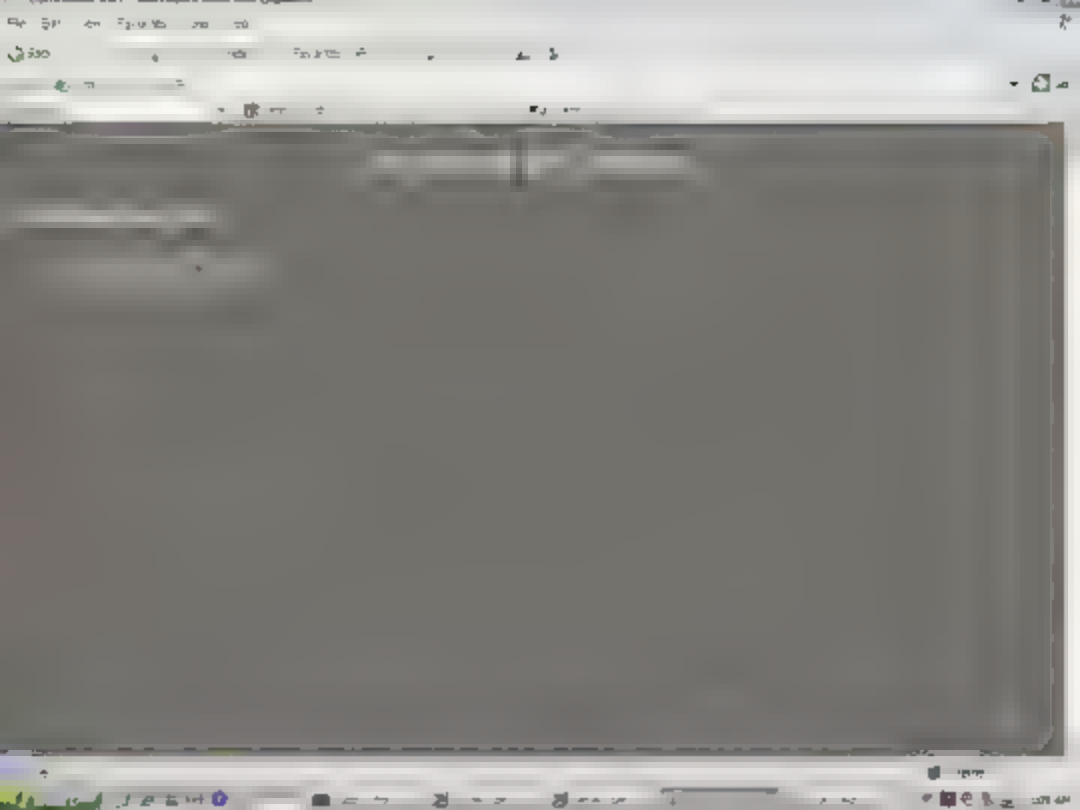
- Introduction
- Singularity OS
- Language, compiler & runtime
- Opportunities

- [illegible]
- [illegible]
  - [illegible]
- [illegible]
  - [illegible]
    - [illegible]
    - [illegible]
  - [illegible]
    - [illegible]
  - [illegible]
    - [illegible]
    - [illegible]
  - [illegible]
    - [illegible]

- [illegible]
- [illegible]
- [illegible]
- [illegible]
  - [illegible]
    - [illegible]
    - [illegible]
    - [illegible]
  - [illegible]
    - [illegible]


Amazon.com DB
Amazon.com customer


Internet
Singularity extension server
web server
credit card cloaking
web service proxy
Third Party Extension



In Search of Caffeine

Looks like you have quite a few choices.

- Starbucks
- Seattle's Best Coffee
- Tully's
- Tully's web version

Take me to Redmond instead

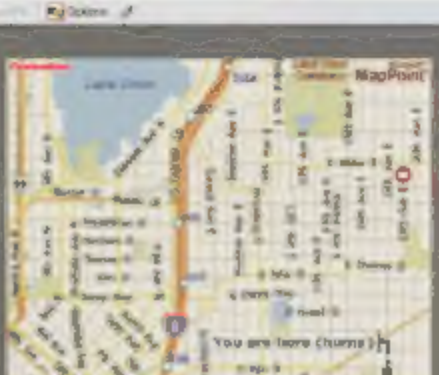
In Search of Caffeine

Looks like you have quite a few choices.

- Tully's
- Tully's with wireless

[Show all three]

[Take me to Redmond instead]



Conclusion

- Singularity enables new systems and applications
  - started with: "what would a system that favored reliability over performance look like?"
  - building platform, tools, and technology to achieve this goal of improving software reliability
  - resulting system enables new software delivery vehicles and businesses
- Open research questions
  - verifiable programming abstractions
  - increased typed assembly language expressiveness
  - general error-handling and recovery mechanisms
  - more specification and better static defect detection tools
  - simpler, robust security model
  - first class application abstraction

Questions?

- http://singularity